

This is the tutorial for IBM Debug Tool for z/OS[®], one of the IBM zSeries[®] problem determination tools.

Using Debug Tool's terminal interface

- Full-screen windows and navigation
- Using the debugger



- Stepping through statements and running the program
- Program statement breakpoints
- Monitoring variables
- Displaying variables in the log
- Making breakpoints conditional
- Variable change breakpoints
- Program entry and exit breakpoints
- Jumping to a statement
- Ending the debugging session

This is the first of three sections that describes how to use the debugger.

This section will cover the first two topics, stepping through statements and running the program, and using program statement breakpoints.

The debugger starts at the top of the program



```
COBOL  LOCATION: SAM1 initialization
Command ==>
Scroll ==> PAGE
MONITOR +---1---+---2---+---3---+---4---+---5---+---6- LINE: 0 OF 0
***** TOP OF MONITOR *****
***** BOTTOM OF MONITOR *****

SOURCE: SAM1 +---1---+---2---+---3---+---4---+---5---+---6- LINE: 1 OF 467
1 *****
2 * PROGRAM: SAM1
3 * Sample program for the ENTERPRISE COBOL Compiler
4 *
5 * AUTHOR : Doug Stout
6 * IBM PD TOOLS

LOG 0 +---1---+---2---+---3---+---4---+---5---+---6- LINE: 32 OF 34
0032 IBM Debug Tool Version 10 Release 1 Mod 0
0033 12/23/2009 6:37:17 PM
0034 5655-V50: Copyright IBM Corp. 1992, 2009
PF 1: ? 2: STEP 3: QUIT 4: LIST 5: FIND 6: AT/CLEAR
PF 7: UP 8: DOWN 9: GO 10: ZOOM 11: ZOOM LOG 12: RETRIEVE
```

The program is not yet initialized

In this example, a Debug Tool session has started, and the 3270 interface is displayed on a terminal. At the very beginning of the session, the program is not yet initialized. In some cases, you may see a message in the log window, saying that there was an error when the debugger tried to open the preferences file or log file. These are files that Debug Tool tries to open automatically when it starts. Unless you specifically intended for Debug Tool to use one of these optional files, you can ignore these startup messages.

Use a STEP command to run one statement at a time



```
COBOL  LOCATION: SAM1 initialization
Command ==> STEP                               Scroll ==> PAGE
MONITOR +-----2-----3-----4-----5-----6- LINE: 0 OF 0
***** TOP OF MONITOR *****
***** BOTTOM OF MONITOR *****

SOURCE: SAM1 +---1---+---2---+---3---+---4---+---5---+ LINE: 1 OF 467
1 *****
2 * PROGRAM: SAM1
3 * Sample program for the ENTERPRISE COBOL Compiler
4 *
5 * AUTHOR : Doug Stout
6 * IBM PD TOOLS

LOG 0-----1-----2-----3-----4-----5----- LINE: 32 OF 34
0032 IBM Debug Tool Version 10 Release 1 Mod 0
0033 12/23/2009 6:37:17 PM
0034 5655-V50: Copyright IBM Corp. 1992, 2009
PF 1:?      2:STEP    3:QUIT    4:LIST    5:FIND    6:PAUSE
PF 7:UP     8:DOWN    9:GO     10:ZOOM   11:ZOOM LOG 12:PAGE
```



You can run your program one statement at a time using the step command. STEP is typed on the command line, and the Enter key is pressed.

After one STEP



```
COBOL LOCATION: SAM1 ENTRY
Command ==>
Scroll ==> PAGE
MONITOR +---1---+---2---+---3---+---4---+---5---+---6- LINE: 0 OF 0
***** TOP OF MONITOR *****
***** BOTTOM OF MONITOR *****

SOURCE: SAM1 +---1---+---2---+---3---+---4---+---5--- LINE: 31 OF 467
31 PROGRAM-ID, SAM1.
32 ENVIRONMENT DIVISION.
33 INPUT-OUTPUT SECTION.
34 FILE-CONTROL.
35
36 SELECT CUSTOMER-FILE ASSIGN TO CUSTFILE

LOG 0 +---1---+---2---+---3---+---4---+---5--- LINE: 33 OF 35
0033 12/23/2009 6:37:17 PM
0034 5655-V50: Copyright IBM Corp.
0035 STEP ;
PF 1: ? 2: STEP 3: QUIT 4: LIST 5: FIND 6: P
PF 7: UP 8: DOWN 9: GO 10: ZOOM 11: ZOOM LOG 12: R
```

The current position

The default F2 key is STEP

F2

5 IBM Debug Tool for z/OS tutorial © 2012 IBM Corporation

That stepped into the program. The red line indicates the current statement in the source window. Notice that the F2 key is set to step by default. F2 is pressed.

After a STEP command



```
COBOL LOCATION: SAM1 :> 251.1
Command ==>
Scroll ==> PAGE
MONITOR --+---1---+---2---+---3---+---4---+---5---+---6- LINE: 0 OF 0
***** TOP OF MONITOR *****
***** BOTTOM OF MONITOR *****

SOURCE: SAM1 +---1---+---2---+---3---+---4---+---5--- LINE: 249 OF 467
249
250 000-MAIN.
251 ACCEPT CURRENT-DATE FROM DATE.
252 ACCEPT CURRENT-TIME FROM TIME.
253 DISPLAY 'SAM1 STARTED DATE = ' CURRENT-MONTH '/'
254 CURRENT-DAY '/' CURRENT-YEAR ' (mm/dd/yy) '.

LOG 0---+---1---+---2---+---3---+---4---+---5---+---6- LINE: 34 OF 36
0034 5655-V50: Copyright IBM Corp. 1992, 2009
0035 STEP ;
0036 STEP ;
PF 1: ? 2: STEP 3: QUIT 4: LIST 5: FIND 6: F
PF 7: UP 8: DOWN 9: GO 10: ZOOM 11: ZOOM LOG 12: R
```

The red line indicates the current statement. It has not executed yet, and will be the next statement to execute.

Commands are written to the log.

F8

6 IBM Debug Tool for z/OS tutorial © 2012 IBM Corporation

One more statement ran, and the red line moved. You can continue using the step command or F2 to run a statement at a time. Notice the contents of the log window now. The step commands were logged automatically. Command logging can be helpful, since you can see the commands you issued so far to run the program to its current location.

The F8 key is pressed to scroll forward in the source window.

An AT command sets a breakpoint



```
COBOL LOCATION: SAM1 :> 251.1
Command ==> AT 258
Scroll ==> PAGE
MONITOR +-----1-----2-----3-----4-----5-----6- LINE: 0 OF 0
***** TOP OF MONITOR *****
***** BOTTOM OF MONITOR *****

This will set a breakpoint
at statement 258

SOURCE: SAM1 +-----1-----2-----3-----4-----5--- LINE: 255 OF 467
255 DISPLAY ' TIME = ' CURRENT-HOUR ':'
256 CURRENT-MINUTE ':' CURRENT-SECOND.
257
258 PERFORM 900-OPEN-TRAN-AND-RPT-FILES.
259 PERFORM 800-INIT-REPORT .
260

LOG 0-----1-----2-----3-----4-----5----- LINE: 34 OF 36
0034 5655-W70: Copyright IBM Corp. 1992, 2012
0035 STEP ;
0036 STEP ;
PF 1: ?      2: STEP      3: QUIT      4: LIST      5: FIND      6: F
PF 7: UP     8: DOWN     9: GO       10: ZOOM     11: ZOOM LOG  12: R
```

"AT 258" is typed on the command line, which will set a breakpoint at statement number 258. You will see some different ways to set breakpoints later. Enter is pressed.

A GO command runs the program



```
COBOL  LOCATION: SAM1 :> 251.1
Command ==> GO                               Scroll ==> PAGE
MONITOR -----2-----3-----4-----5-----6- LINE: 0 OF 0
***** TOP OF MONITOR *****
***** BOTTOM OF MONITOR *****

SOURCE: SAM1 +---1---+---2---+---3---+---4---+---5--- LINE: 255 OF 467
255 DISPLAY '          TIME = ' CURRENT-HOUR ':'
256          CURRENT-MINUTE ':' CURRENT-SECOND.
257
258          PERFORM 900-OPEN-TRAN-AND-RPT-FILES.
259          PERFORM 800-INIT-REPORT .
260

LOG 0-----1-----2-----3-----4-----5----- LINE: 35 OF 37
0035 STEP ;
0036 STEP ;
0037 AT 258 ;
PF 1: ?      2: STEP      3: QUIT      4: LIST      5: FIND      6: PAUSE
PF 7: UP     8: DOWN     9: GO       10: ZOOM     11: ZOOM LOG  12: PAGE

8 | IBM Debug Tool for z/OS tutorial | © 2012 IBM Corporation
```

A breakpoint was set on line 258. Notice that the line number for that statement is highlighted now, which indicates that a breakpoint has been set there.

A GO command is entered. Use GO commands to run the program. Enter.

Result of a GO command - the program ran until it reached a breakpoint



```
COBOL LOCATION: SAM1 :> 258.1
Command ==> █ Scroll ==> PAGE
MONITOR --+---1---+---2---+---3---+---4---+---5---+---6- LINE: 0 OF 0
***** TOP OF MONITOR *****
***** BOTTOM OF MONITOR *****

SOURCE: SAM1 +---1---+---2---+---3---+---4---+---5--- LINE: 255 OF 467
255 DISPLAY ' TIME = ' CURRENT-HOUR ':'
256 CURRENT-MINUTE ':' CURRENT-SECOND.
257
258 PERFORM 900-OPEN-TRAN-AND-RPT-FILES.
259 PERFORM 800-INIT-REPORT .
260

LOG 0---+---1---+---2---+---3---+---4---+---5---+--- LINE: 36 OF 38
0036 STEP ;
0037 AT 258 ;
0038 GO ;
PF 1: ? 2: STEP 3: QUIT 4: LIST 5: FIND
PF 7: UP 8: DOWN 9: GO 10: ZOOM 11: ZOOM LOG 12:

Note: the default F9 key is GO

F8
F8
```

9

IBM Debug Tool for z/OS tutorial

© 2012 IBM Corporation

The program ran until it encountered a breakpoint. Use statement breakpoints at strategic places in your program where you want to pause. When a statement breakpoint triggers, the program pauses before the statement runs. As you saw, setting a breakpoint followed by a GO command is one way to run to a statement.

And there is another way to quickly run to a statement. First, the F8 key is pressed a few times to scroll down.

A RUNTO *line-number* command runs the program until it reaches the statement



```
COBOL  LOCATION: SAM1 :> 258.1
Command ==> RUNTO 275
Scroll ==> PAGE
MONITOR --+-----1-----2-----3-----4-----5-----6- LINE: 0 OF 0
***** TOP OF MONITOR *****
***** BOTTOM OF MONITOR *****

SOURCE: SAM1 +---1---+---2---+---3---+---4---+---5--- LINE: 274 OF 467
274      WRITE REPORT-RECORD FROM RPT-TRAN-DETAIL
275      MOVE 'Y' TO WS-TRAN-OK
276      EVALUATE TRAN-CODE
277          WHEN 'PRINT '
278          PERFORM 200-PROCESS-PRINT-TRAN
279          WHEN 'TOTALS'

LOG 0-----1-----2-----3-----4-----5----- LINE: 36 OF 38
0036 STEP ;
0037 AT 258 ;
0038 GO ;
PF 1: ?      2: STEP      3: QUIT      4: LIST      5: FIND      6: PAUSE
PF 7: UP     8: DOWN     9: GO      10: ZOOM     11: ZOOM LOG  12: PAGE
```



You can use a RUNTO command to run the program until it reaches a specific statement number. The command "RUNTO 275" is entered.

Result of a RUNTO 275 command



```
COBOL LOCATION: SAM1 :> 275.1
Command ==> █ Scroll ==> PAGE
MONITOR +-----1-----2-----3-----4-----5-----6- LINE: 0 OF 0
***** TOP OF MONITOR *****
***** BOTTOM OF MONITOR *****

The program ran until it reached
the RUNTO statement

SOURCE: SAM1 +-----1-----2-----3-----4-----5--- LINE: 274 OF 467
274 WRITE REPORT-RECORD FROM RPT-TRAN-DETAIL
275 MOVE 'Y' TO WS-TRAN-OK
276 EVALUATE TRAN-CODE
277 WHEN 'PRINT '
278 PERFORM 200-PROCESS-PRINT-TRAN
279 WHEN 'TOTALS'
LOG 0-----1-----2-----3-----4-----5----- LINE: 37 OF 39
0037 AT 258 ;
0038 GO ;
0039 RUNTO 275 ;
PF 1: ? 2: STEP 3: QUIT 4: LIST 5: FIND 6: AT/CLEAR
PF 7: UP 8: DOWN 9: GO 10: ZOOM 11: ZOOM LOG 12: RETRIEVE
```

11

IBM Debug Tool for z/OS tutorial

© 2012 IBM Corporation

And the program ran until it reached statement 275. The red line is on 275, which means that it is the current line. The program is paused before line 275 has run.

An R line command is the same as a RUNTO command



```
COBOL LOCATION: SAM1 :> 275.1
Command ==> █ Scroll ==> PAGE
MONITOR +-----1-----2-----3-----4-----5-----6- LINE: 0 OF 0
***** TOP OF MONITOR *****
***** BOTTOM OF MONITOR *****

SOURCE: SAM1 +-----1-----2-----3-----4-----5--- LINE: 274 OF 467
274 WRITE REPORT-RECORD FROM RPT-TRAN-DETAIL
275 MOVE 'Y' TO WS-TRAN-OK
276 EVALUATE TRAN-CODE
277 WHEN 'PRINT '
R 278 PERFORM 200-PROCESS-PRINT-TRAN
279 WHEN 'TOTALS'

LOG 0-----1-----2-----3-----4-----5----- LINE: 37 OF 39
0037 AT 258 ;
0038 GO ;
0039 RUNTO 275 ;
PF 1: ? 2: STEP 3: QUIT 4: LIST 5: FIND 6: PAUSE
PF 7: UP 8: DOWN 9: GO 10: ZOOM 11: ZOOM LOG 12: R
```



12

IBM Debug Tool for z/OS tutorial

© 2012 IBM Corporation

An R line command is the same as a RUNTO command. An R line command is entered next to line 278, and Enter is pressed.

Result of an R line command on statement 278



```
COBOL LOCATION: SAM1 :> 278.1
Command ==>
Scroll ==> PAGE
MONITOR +-----1-----2-----3-----4-----5-----6- LINE: 0 OF 0
***** TOP OF MONITOR *****
***** BOTTOM OF MONITOR *****

The program ran until it reached
the RUNTO statement

SOURCE: SAM1 +-----1-----2-----3-----4-----5--- LINE: 274 OF 467
274 WRITE REPORT-RECORD FROM RPT-TRAN-DETAIL
275 MOVE 'Y' TO WS-TRAN-OK
276 EVALUATE TRAN-CODE
277 WHEN 'PRINT '
278 PERFORM 200-PROCESS-PRINT-TRAN
279 WHEN 'TOTALS'

LOG 0-----1-----2-----3-----4-----5----- LINE: 38 OF 40
0038 GO ;
0039 RUNTO 275 ;
0040 RUNTO 278 ;
PF 1: ? 2: STEP 3: QUIT 4: LIST 5: FIND 6: AT/CLEAR
PF 7: UP 8: DOWN 9: GO 10: ZOOM 11: ZOOM LOG 12: RETRIEVE
```

And the program ran to line 278.

Commands to run the program STEP and GO



- STEP or the default F2 key
 - Runs one statement and then stops
 - A frequently used command option:
 - STEP n performs *n* steps
 - For example, STEP 20 displays 20 steps with a brief pause between each one. After 20 steps, it stops.

- GO or the default F9 key
 - Runs the program
 - The program stops when it reaches:
 - a breakpoint, or
 - a condition (such as an abend)
 - If a breakpoint or condition is not encountered, the application runs to completion
 - The application ends and the debugging session ends

You can use a step command, or the default F2 key, to run one statement at a time. You can also use a feature called step animation. For example, you could enter the command: "step 20" on the command line. When you press enter, you will see the program run the next 20 statements automatically one at a time. It is like watching a movie; you can sit back and watch the program run. Sometimes step animation is helpful if your program is in a tight loop, and you want to see it iterate through the loop.

To run the program, use a GO command or the F9 key. To stop at a statement, set a breakpoint there, and issue a GO command. The program will stop when it reaches a breakpoint, or a program condition, such as an abend. Use some caution though, because if a breakpoint or condition is not encountered as the program runs, the application runs to completion and the debugging session ends.

Commands to run the program

RUNTO



- RUNTO *statement-number* , or
- R line command in the source window
 - Runs the program until it reaches *statement-number*
 - Use it to stop at a statement without having to set a breakpoint
 - The program stops if it reaches:
 - *statement-number*, or
 - a breakpoint, or
 - a condition (such as an abend)
 - If *statement-number* is not reached:
 - the RUNTO remains active for subsequent GO and RUNTO commands
 - If *statement-number*, a breakpoint, or condition is not encountered, the application runs to completion
 - The application ends and the debugging session ends

There is an even simpler way to run to a statement, the RUNTO command. For example, if you type in the command “RUNTO 500”, the program will run and pause the next time it gets to statement 500. That way, you do not even have to set a breakpoint first. The R line command is the same as a RUNTO command.

With a RUNTO command or R line command, the program stops when it reaches the statement, another breakpoint, or a program condition (such as an abend), which ever is encountered first.

If the statement was not reached, the RUNTO remains active for subsequent GO and RUNTO commands.

If the statement, a breakpoint, or a condition is not encountered, the application runs to completion, the application terminates and the debugging session ends.

Using Debug Tool's terminal interface

- Full-screen windows and navigation
- Using the debugger
 - Stepping through statements and running the program
 - Program statement breakpoints
 - Monitoring variables
 - Displaying variables in the log
 - Making breakpoints conditional
 - Variable change breakpoints
 - Program entry and exit breakpoints
 - Jumping to a statement
 - Ending the debugging session



Next, you will see how to set breakpoints that will pause the program when it reaches specific statements.

A ZOOM command displays one window full-screen



The screenshot shows the IBM Debug Tool for z/OS interface. At the top, the command 'ZOOM' is entered and highlighted with a red circle. Below the command, the 'MONITOR' window is visible, showing the source code for 'SOURCE: SAM1'. The source code includes lines 273 through 278, with line 278 highlighted in red. A yellow callout box points to the source window with the text 'The source window zooms by default'. At the bottom right, a yellow box labeled 'Enter' indicates the key used to execute the command. The interface also shows a 'LOG' window at the bottom with various commands and a keyboard shortcut menu.

```
COBOL LOCATION: SAM1 :> 278.1
Command ==> ZOOM |
Scroll ==> PAGE
MONITOR --+---1---+---2---+---3---+---4---+---5---+---6- LINE: 0 OF 0
***** TOP OF MONITOR *****
***** BOTTOM OF MONITOR *****

SOURCE: SAM1 +---1---+---2---+---3---+---4---+---5--- LINE: 273 OF 467
273 MOVE TRAN-RECORD TO RPT-TRAN-RECORD
274 WRITE REPORT-RECORD FROM RPT-TRAN-RECORD
275 MOVE 'Y' TO WS-TRAN-OK
276 EVALUATE TRAN-CODE
277 WHEN 'PRINT '
278 PERFORM 200-PROCESS-PRINT-TRAN

LOG 0---+---1---+---2---+---3---+---4---+---5---+--- LINE: 38 OF 40
0038 GO ;
0039 RUNTO 275 ;
0040 RUNTO 278 ;
PF 1: ?      2: STEP      3: QUIT      4: LIST      5: FIND      6: PAGE
PF 7: UP     8: DOWN     9: GO       10: ZOOM     11: ZOOM LOG  12: PAGE
```

Remember that you can use a zoom command or the F10 key to zoom in on a window. Here, a ZOOM command is entered to zoom in on the source window.

A POS line-number command positions to a source statement



```
COBOL LOCATION: SAM1 :> 251.1
Command ==<> POS 306 Scroll ==> PAGE
SOURCE: SAM1 +-----2-----3-----4-----5--- LINE: 249 OF 467
249
250          000-MAIN.
251          ACCEPT CURRENT-DATE FROM DATE.
252          ACCEPT CURRENT-TIME FROM TIME.
253          DISPLAY 'SAM1 STARTED DATE = ' CURRENT-MONTH '/'
254             CURRENT-DAY '/' CURRENT-YEAR ' (mm/dd/yy) '.
255          DISPLAY '          TIME = ' CURRENT-HOUR ':'
256             CURRENT-MINUTE ':' CURRENT-SECOND.
257
258          PERFORM 900-OPEN-TRAN-AND-RPT-FILES.
259          PERFORM 800-INIT-REPORT .
260
261          PERFORM 100-PROCESS-TRANSACTIONS
262             UNTIL WS-TRAN-FILE-EOF = 'Y' .
263
264          PERFORM 905-CLOSE-TRAN-AND-RPT-FILES.
265
266          GOBACK .
267
PF 1: ?      2: STEP      3: QUIT      4: LIST      5: FIND      6: A
PF 7: UP     8: DOWN     9: GO       10: ZOOM    11: ZOOM LOG 12: R
```



If you are working on a large program, it can be helpful to quickly position to a particular place in the program. Use the POSition command to scroll the source window to a statement. The command "POS 306" is entered.

Result of a POS 306 command



```
COBOL      LOCATION: SAM1 :> 278.1
Command ==>
Scroll ==> PAGE
SOURCE: SAM1 +-----1-----2-----3-----4-----5--- LINE: 306 OF 467
306      210-PROCESS-CUSTFILE-RECORD.
307      PERFORM 730-READ-CUSTOMER-FILE.
308      IF WS-CUST-FILE-EOF NOT = 'Y'
309      IF CUST-RECORD-TYPE = 'C'
310      ADD +1 TO NUM-CUSTOMER-RECS
311      *
          SUBROUTINE SAM2 WILL COLLECT CUSTOMER STATISTICS
          CALL 'SAM2' USING CUST-REC,
              CUSTOMER-BALANCE-STATS
315      MOVE CUST-ID          TO RPT-CUST-ID
316      MOVE CUST-NAME        TO RPT-CUST-NAME
317      MOVE CUST-OCCUPATION  TO RPT-CUST-OCCUPATION
318      MOVE CUST-ACCT-BALANCE TO RPT-CUST-ACCT-BALANCE
319      MOVE CUST-ORDERS-YTD  TO RPT-CUST-ORDERS-YTD
320      WRITE REPORT-RECORD FROM RPT-DETAIL AFTER 1
321      ADD +1 TO NUM-DETAIL-LINES
322      END-IF
323      IF CUST-RECORD-TYPE = 'P'
324      *
          SUBROUTINE SAM3 WILL COLLECT PRODUCT STATISTICS
          CALL 'SAM3' USING CUST-REC,

```

Positioned to statement 306

```
PF 1: ?      2: STEP      3: QUIT      4: LIST      5: FIND      6: AT/CLEAR
PF 7: UP     8: DOWN     9: GO      10: ZOOM     11: ZOOM LOG  12: RETRIEVE
```

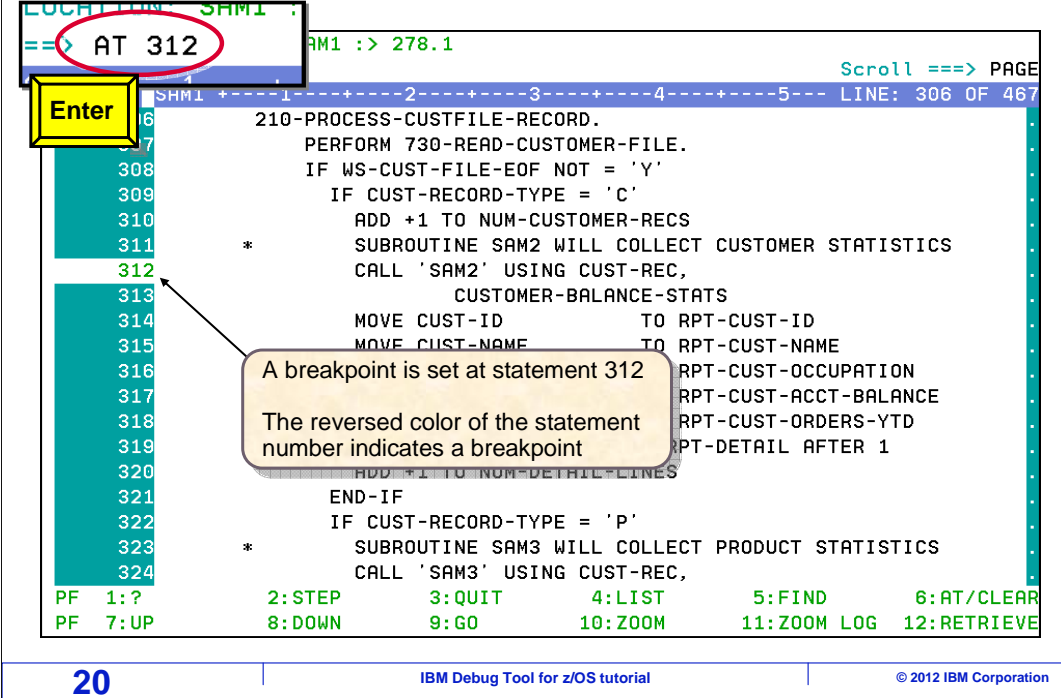
19

IBM Debug Tool for z/OS tutorial

© 2012 IBM Corporation

That positioned the source window to statement 306.

An AT line-number command sets a statement breakpoint



```
LOCATION SAM1 :
==> AT 312 SAM1 :> 278.1
Scroll ==> PAGE
SAM1 +-----1-----2-----3-----4-----5--- LINE: 306 OF 467
306      210-PROCESS-CUSTFILE-RECORD.
307      PERFORM 730-READ-CUSTOMER-FILE.
308      IF WS-CUST-FILE-EOF NOT = 'Y'
309      IF CUST-RECORD-TYPE = 'C'
310      ADD +1 TO NUM-CUSTOMER-RECS
311      * SUBROUTINE SAM2 WILL COLLECT CUSTOMER STATISTICS
312      CALL 'SAM2' USING CUST-REC,
313      CUSTOMER-BALANCE-STATS
314      MOVE CUST-ID          TO RPT-CUST-ID
315      MOVE CUST-NAME       TO RPT-CUST-NAME
316      RPT-CUST-OCCUPATION
317      RPT-CUST-ACCT-BALANCE
318      RPT-CUST-ORDERS-YTD
319      RPT-DETAIL AFTER 1
320      ADD +1 TO NUM-DETAIL-LINES
321      END-IF
322      IF CUST-RECORD-TYPE = 'P'
323      * SUBROUTINE SAM3 WILL COLLECT PRODUCT STATISTICS
324      CALL 'SAM3' USING CUST-REC,
PF 1: ?      2: STEP      3: QUIT      4: LIST      5: FIND      6: AT/CLEAR
PF 7: UP     8: DOWN      9: GO       10: ZOOM     11: ZOOM LOG  12: RETRIEVE
```

A breakpoint is set at statement 312
The reversed color of the statement number indicates a breakpoint

20 IBM Debug Tool for z/OS tutorial © 2012 IBM Corporation

Next, you will see three ways to set a statement breakpoint. First, you can use an AT command. The command "AT 312" is typed on the command line, and Enter is pressed. That set a breakpoint at line 312. The line number for the statement is highlighted, which indicates that a breakpoint has been set.

The AT/CLEAR key sets or clears a breakpoint at the statement selected by your cursor



```
COBOL      LOCATION: SAM1 :> 278.1
Command ==>                               Scroll ==> PAGE
SOURCE: SAM1 +----1-----2-----3-----4-----5--- LINE: 306 OF 467
306      210-PROCESS-CUSTFILE-RECORD.
307      PERFORM 730-READ-CUSTOMER-FILE.
308      IF WS-CUST-FILE-EOF NOT = 'Y'
309      IF CUST-RECORD-TYPE = 'C'
310      ADD +1 TO NUM-CUSTOMER-RECS
311      *      SUBROUTINE SAM2 WILL COLLECT CUSTOMER STATISTICS
312      CALL 'SAM2' USING CUST-REC,
313      CUSTOMER-BALANCE-STATS
314      MOVE CUST-ID          TO RPT-CUST-ID
315      MOVE CUST-NAME        TO RPT-CUST-NAME
316      MOVE CUST-OCCUPATION  TO RPT-CUST-OCCUPATION
317      MOVE CUST-ACCT-BALANCE TO RPT-CUST-ACCT-BALANCE
318      MOVE CUST-ORDERS-YTD  TO RPT-CUST-ORDERS-YTD
319      WRITE REPORT-RECORD FROM RPT-DETAIL AFTER 1
320      ADD +1 TO NUM-DETAIL-LINES
321      END-IF
322      IF CUST-RECORD-TYPE = 'P'
323      *      SUBROUTINE SAM3 WILL COLLECT PRODUCT STATISTICS
324      CALL 'SAM3' USING CUST-REC,
PF 1: ?      2: STEP      3: QUIT      4: LIST      5: FIND      6: AT/CLEAR
PF 7: UP      8: DOWN      9: GO       10: ZOOM     11: ZOOM LOG   12: RETRIEVE
```

21

IBM Debug Tool for z/OS tutorial

© 2012 IBM Corporation

Another way to set a breakpoint at a statement is by using the F6 key. Notice that F6 is defined as AT/CLEAR. F6 is a toggle that will turn a statement breakpoint on or off. To use it, first place your cursor on the statement where you want the breakpoint. In this example, the cursor is put somewhere on statement number 314, and then F6 is pressed.

An A or AT line command sets a statement breakpoint



```
COBOL      LOCATION: SAM1 :> 278.1
Command ==>
SOURCE: SAM1 +----1-----2-----3-----4-----5--- LINE: 306 OF 467
306      210-PROCESS-CUSTFILE-RECORD.
307      PERFORM 730-READ-CUSTOMER-FILE.
308      IF WS-CUST-FILE-EOF NOT = 'Y'
309      IF CUST-RECORD-NO = 0
310      ADD +1 TO NUM-CUSTOMER-RECS
311      *      SUBROUTINE SAM2 WILL COLLECT CUSTOMER STATISTICS
312      CALL 'SAM2' USING CUST-REC,
313      CUSTOMER-BALANCE-STATS
314      MOVE CUST-ID          TO RPT-CUST-ID
315      MOVE CUST-NAME        TO RPT-CUST-NAME
316      MOVE CUST-OCCUPATION  TO RPT-CUST-OCCUPATION
317      MOVE CUST-ACCT-BALANCE TO RPT-CUST-ACCT-BALANCE
318      MOVE CUST-ORDERS-YTD  TO RPT-CUST-ORDERS-YTD
319      WRITE REPORT-RECORD FROM RPT-DETAIL AFTER 1
320      ADD +1 TO NUM-DETAIL-LINES
321      END-IF
322      IF CUST-RECORD-TYPE = 'P'
323      *      SUBROUTINE SAM3 WILL COLLECT PRODUCT STATISTICS
324      CALL 'SAM3' USING CUST-REC,
PF 1: ?      2: STEP      3: QUIT      4: LIST      5: FIND      6: AT ALL
PF 7: UP     8: DOWN     9: GO      10: ZOOM     11: ZOOM LOG  12: RETRIEVE
```

22

IBM Debug Tool for z/OS tutorial

© 2012 IBM Corporation

Now there is a new breakpoint at statement 314. The third way to set a statement breakpoint is by using the A or AT line command. "A" is typed on the line command area of statement 316, and Enter is pressed.

Statement breakpoints were added



```
COBOL      LOCATION: SAM1 :> 278.1
Command ==>
SOURCE: SAM1 +-----1-----2-----3-----4-----5--- LINE: 306 OF 467
306      210-PROCESS-CUSTFILE-RECORD.
307          PERFORM 730-READ-CUSTOMER-FILE.
308          IF WS-CUST-FILE-EOF NOT = 'Y'
309              IF CUST-RECORD-TYPE = 'C'
310                  ADD +1 TO NUM-CUSTOMER-RECS
311          *      SUBROUTINE SAM2 WILL COLLECT CUSTOMER STATISTICS
312          CALL 'SAM2' USING CUST-REC,
313                  CUSTOMER-BALANCE-STATS
314          MOVE CUST-ID          TO RPT-CUST-ID
315          MOVE CUST-NAME        TO RPT-CUST-NAME
316          MOVE CUST-OCCUPATION  TO RPT-CUST-OCCUPATION
317          MOVE CUST-ACCT-BALANCE TO RPT-CUST-ACCT-BALANCE
318          MOVE CUST-ORDERS-YTD  TO RPT-CUST-ORDERS-YTD
319          WRITE REPORT-RECORD FROM RPT-DETAIL AFTER 1
320          ADD +1 TO NUM-DETAIL-LINES
321      END-IF
322      IF CUST-RECORD-TYPE = 'P'
323          *      SUBROUTINE SAM3 WILL COLLECT PRODUCT STATISTICS
324          CALL 'SAM3' USING CUST-REC,
```

PF 1: ? 2: STEP 3: QUIT 4: LIST 5: FIND 6: A
PF 7: UP 8: DOWN 9: GO 10: ZOOM 11: ZOOM LOG 12: R



That set a breakpoint at statement 316. At this point, there are three statement breakpoints set. F10 will zoom out to display all the windows again.

A GO command runs the program



```
COBOL  LOCATION: SAM1 :> 278.1
Command ==> GO                               Scroll ==> PAGE
MONITOR -----2-----3-----4-----5-----6- LINE: 0 OF 0
***** TOP OF MONITOR *****
***** BOTTOM OF MONITOR *****

SOURCE: SAM1 +---1---+---2---+---3---+---4---+---5--- LINE: 273 OF 467
273          MOVE TRAN-RECORD TO RPT-TRAN-RECORD
274          WRITE REPORT-RECORD FROM RPT-TRAN-DETAIL
275          MOVE 'Y' TO WS-TRAN-OK
276          EVALUATE TRAN-CODE
277          WHEN 'PRINT '
278          PERFORM 200-PROCESS-PRINT-TRAN

LOG 0-----1-----2-----3-----4-----5----- LINE: 41 OF 43
0041 AT 312 ;
0042 AT 314 ;
0043 AT 316 ;
PF 1: ?      2: STEP      3: QUIT      4: LIST      5: FIND      6: PAUSE
PF 7: UP     8: DOWN     9: GO      10: ZOOM     11: ZOOM LOG 12: R

Enter
```

A GO command is entered.

The program ran until it reached a breakpoint



```
COBOL LOCATION: SAM1 :> 312.1
Command ==>
Scroll ==> PAGE
MONITOR +-----1-----2-----3-----4-----5-----6- LINE: 0 OF 0
***** TOP OF MONITOR *****
***** BOTTOM OF MONITOR *****

The red line indicates the current
statement. Statement 312 has not
executed yet.

SOURCE: SAM1 +-----1-----2-----3-----4-----5--- LINE: 311 OF 467
311 * SUBROUTINE SAM2 WILL COLLECT CUSTOMER STATISTICS
312 CALL 'SAM2' USING CUST-REC,
313 CUSTOMER-BALANCE-STATS
314 MOVE CUST-ID TO RPT-CUST-ID
315 MOVE CUST-NAME TO RPT-CUST-NAME
316 MOVE CUST-OCCUPATION TO RPT-CUST-OCCUPATION

LOG 0-----1-----2-----3-----4-----5----- LINE: 42 OF 44
0042 AT 314 ;
0043 AT 316 ;
0044 GO ;
PF 1: ? 2: STEP 3: QUIT 4: LIST 5: FIND 6: PAUSE
PF 7: UP 8: DOWN 9: GO 10: ZOOM 11: ZOOM LOG 12: R

The GO function key is the
same as a GO command

F9
```

25 IBM Debug Tool for z/OS tutorial © 2012 IBM Corporation

The program ran until it reached a breakpoint. Use statement breakpoints to pause your program at specific places. When a statement breakpoint is triggered, the program is paused before the statement runs. Notice that F9 is set to the go command. F9 is pressed.

Result of F9 key The program ran to the next breakpoint

```

COBOL      LOCATION: SAM1 :> 314.1
Command ==> █                               Scroll ==> PAGE
MONITOR --+---1---+---2---+---3---+---4---+---5---+---6- LINE: 0 OF 0
***** TOP OF MONITOR *****
***** BOTTOM OF MONITOR *****

SOURCE: SAM1 +---1---+---2---+---3---+---4---+---5--- LINE: 311 OF 467
311      *          SUBROUTINE SAM2 WILL COLLECT CUSTOMER STATISTICS
312      CALL 'SAM2' USING CUST-REC,
313      CUSTOMER-BALANCE-STATS
314      MOVE CUST-ID          TO RPT-CUST-ID
315      MOVE CUST-NAME        TO RPT-CUST-NAME
316      MOVE CUST-OCCUPATION  TO RPT-CUST-OCCUPATION

LOG 0---+---1---+---2---+---3---+---4---+---5---+--- LINE: 43 OF 45
0043 AT 316 ;
0044 GO ;
0045 GO ;
PF 1: ?          2: STEP          3: QUIT          4: LIST          5: FIND          6: AT/CLEAR
PF 7: UP         8: DOWN          9: GO            10: ZOOM         11: ZOOM LOG     12: RETRIEVE

```

And the program ran until it came to the next breakpoint. When you are stopped at a breakpoint, you can examine and change program variable values, set other breakpoints, and remove breakpoints.

A LIST AT command displays a list of breakpoints



The screenshot shows the IBM Debug Tool interface. At the top, the command `LIST AT` is entered in a command window, with a yellow box labeled "Enter" below it. The main display area shows the source code for a subroutine named SAM2, with line 314 highlighted in red. Below the source code, the log window displays the output of the `LIST AT` command, showing a list of breakpoints. A yellow box labeled "The breakpoints are displayed in the log" points to this output. At the bottom of the log window, a keymap shows the `F11` key assigned to the `11: ZOOM LOG` function, with a yellow box labeled "F11" and a callout "Zoom the log window" pointing to it. The footer of the screen displays the page number `27`, the text `IBM Debug Tool for z/OS tutorial`, and the copyright notice `© 2012 IBM Corporation`.

Use a "LIST AT" command to display a list of all currently set breakpoints. The list is displayed in the log window. In this example, the list was too big to see it all in the log window.

Notice that the F11 key is set to "zoom log". F11 is pressed.

Zoomed to the log window, result of F11



```
COBOL LOCATION: SAM1 :> 314.1
Command ==>
LOG 0-----1-----2-----3-----4-----5----- LINE: 36 OF 54
0036 STEP ;
0037 AT 258 ;
0038 GO ;
0039 RUNTO 275 ;
0040 RUNTO 278 ;
0041 AT 312 ;
0042 AT 314 ;
0043 AT 316 ;
0044 GO ;
0045 GO ;
0046 LIST AT ;
0047 The STATEMENT SAM1 ::> SAM1 :> 258.1 breakpoint action is:
0048 ;
0049 The STATEMENT SAM1 ::> SAM1 :> 312.1 breakpoint action is:
0050 ;
0051 The STATEMENT SAM1 ::> SAM1 :> 314.1 breakpoint action is:
0052 ;
0053 The STATEMENT SAM1 ::> SAM1 :> 316.1 breakpoint action is:
0054 ;
PF 1:?          2:STEP      3:QUIT      4:LIST      5:FIND      6:A
PF 7:UP         8:DOWN      9:GO       10:ZOOM     11:ZOOM LOG 12:R
```

Commands up to this point

There are currently four breakpoints.
Statement breakpoints are displayed in the format:
LoadModule ::> csect, block, or source :> statement#.verb#

F10

28 IBM Debug Tool for z/OS tutorial © 2012 IBM Corporation

That zoomed in on the log window, and now the full result of the "list at" command can be seen. There are currently four breakpoints set, at statements 258, 312, 314, and 316. The F10 key is pressed to zoom back out.

A CLEAR AT line-number command clears a statement breakpoint



```
LOC: 314.1
==> CLEAR AT 316
Enter
Scroll ==> PAGE
***** TOP OF MONITOR *****
***** BOTTOM OF MONITOR *****

SOURCE: SAM1 +---1---+---2---+---3---+---4---+---5--- LINE: 311 OF 467
311 * SUBROUTINE SAM2 WILL COLLECT CUSTOMER STATISTICS
312 CALL 'SAM2' USING CUST-REC,
313 CUSTOMER-BALANCE-STATS
314 MOVE CUST-ID TO RPT-CUST-ID
315 MOVE CUST-NAME TO RPT-CUST-NAME
316 MOVE CUST-OCCUPATION TO RPT-CUST-OCCUPATION
LOG 0 +---1---+---2---+---3---+---4---+---5--- LINE: 53 OF 55
0053 The STATEMENT SAM1 ::> SAM1 :> 316.1 breakpoint action is:
0054 ;
0055 CLEAR AT 316 ;
PF 1:? 2:STEP 3:QUIT 4:LIST 5:FIND 6:AT/CLEAR
PF 7:UP 8:DOWN 9:GO 10:ZOOM 11:ZOOM LOG 12:RETRIEVE
```

The breakpoint at 316 was cleared

And all three windows are displayed again. You have seen how to set breakpoints at statements and run the program until it reaches your breakpoints. Next you will see how to remove breakpoints. Remember, there are three ways to set breakpoints. And there are three ways to remove them. First, you can enter a command like “CLEAR AT 316” to clear the breakpoint at statement 316. Notice the syntax of the clear command. It is the AT command with the word clear in front of it. The breakpoint at statement 316 was removed.

The AT/CLEAR function key clears a statement breakpoint selected by your cursor



```

COBOL    LOCATION: SAM1 :> 314.1
Command ==>                               Scroll ==> PAGE
MONITOR --+---1---+---2---+---3---+---4---+---5---+---6- LINE: 0 OF 0
***** TOP OF MONITOR *****
***** BOTTOM OF MONITOR *****

SOURCE: SAM1 +---1---+---2---+---3---+---4---+---5--- LINE: 311 OF 467
311      * SUBROUTINE SAM2 WILL COLLECT CUSTOMER STATISTICS
312      CALL 'SAM2' USING CUST-REC,
313      CUSTOMER-BALANCE-STATS
314      MOVE CUST-ID          TO RPT-CUST-ID
315      MOVE CUST-NAME        TO RPT-CUST-NAME
316      MOVE CUST-OCCUPATION  TO RPT-CUST-OCCUPATION
LOG 0---+---1---+---2---+---3---+---4---+---5---+---6- OF 55
0053 The STATEMENT SAM1 ::> SAM1 :> 316.1 breakpoint action is:
0054 ;
0055 CLEAR AT 316 ;
PF 1:?      2:STEP      3:QUIT      4:LIST      5:FIND      6:AT/CLEAR
PF 7:UP     8:DOWN     9:GO      10:ZOOM     11:ZOOM LOG  12:RETRIEVE
  
```

30

IBM Debug Tool for z/OS tutorial

© 2012 IBM Corporation

Another way to remove a breakpoint is with the F6 key. F6 is a toggle, and sets or removes breakpoints. Put your cursor on a statement that already has a breakpoint, and press F6 to remove it. In this example, the cursor is placed somewhere on statement 312, and F6 is pressed.

A C line command clears a statement breakpoint



```
COBOL LOCATION: SAM1 :> 314.1
Command ==> Scroll ==> PAGE
MONITOR --+---1---+---2---+---3---+---4---+---5---+---6- LINE: 0 OF 0
***** TOP OF MONITOR *****
***** BOTTOM OF MONITOR *****

SOURCE: SAM1 +---1---+---2---+---3---+---4---+---5--- LINE: 311 OF 467
311 * SUBROUTINE SAM2 WILL COLLECT CUSTOMER STATISTICS
312 CALL 'SAM2' USING CUST-REC,
313 CUSTOMER-BALANCE-STATS
C 314 MOVE CUST-ID TO RPT-CUST-ID
315 MOVE CUST-NAME TO RPT-CUST-NAME
316 MOVE CUST-OCCUPATION TO RPT-CUST-OCCUPATION
LOG 0--1---+---2---+---3---+---4---+---5---+---6- LINE: 54 OF 56
0054 ;
0055 CLEAR AT 316 ;
0056 CLEAR AT 312 ;
PF 1:? 2:STEP 3:QUIT 4:LIST 5:FIND 6:PAUSE
PF 7:UP 8:DOWN 9:GO 10:ZOOM 11:ZOOM LOG 12:PAGE
```

The breakpoint at 312 was cleared

Enter

31

IBM Debug Tool for z/OS tutorial

© 2012 IBM Corporation

That cleared the breakpoint at statement 312.

The other way to clear a statement breakpoint is with a C line command. "C" is typed as a line command next to statement 314, and Enter is pressed.

Result of a C line command



```
COBOL LOCATION: SAM1 :> 314.1
Command ==>
Scroll ==> PAGE
MONITOR --+---1---+---2---+---3---+---4---+---5---+---6- LINE: 0 OF 0
***** TOP OF MONITOR *****
***** BOTTOM OF MONITOR *****

SOURCE: SAM1 +---1---+---2---+---3---+---4---+---5--- LINE: 311 OF 467
311 * SUBROUTINE SAM2 WILL COLLECT CUSTOMER STATISTICS
312 CALL 'SAM2' USING CUST-REC,
313 CUSTOMER-BALANCE-STATS
314 MOVE CUST-ID TO RPT-CUST-ID
315 MOVE CUST-NAME TO RPT-CUST-NAME
316 MOVE CUST-OCCUPATION TO RPT-CUST-OCCUPATION
LOG 0--1---+---2---+---3---+---4---+---5---+---6- LINE: 55 OF 57
0055 CLEAR AT 316 ;
0056 CLEAR AT 312 ;
0057 CLEAR AT 314 ;
PF 1:? 2:STEP 3:QUIT 4:LIST 5:FIND 6:AT/CLEAR
PF 7:UP 8:DOWN 9:GO 10:ZOOM 11:ZOOM LOG 12:RETRIEVE
```

The breakpoint at 314 was cleared



And the breakpoint at statement 314 was cleared.

A LIST AT command displays a list of breakpoints in the log



```
LOCATION: SAM1
=> LIST AT
Enter
SAM1 :> 314.1
Scroll ==> PAGE
***** TOP OF MONITOR *****
***** BOTTOM OF MONITOR *****

SOURCE: SAM1 +---1---+---2---+---3---+---4---+---5--- LINE: 311 OF 467
311 * SUBROUTINE SAM2 WILL COLLECT CUSTOMER STATISTICS
312 CALL 'SAM2' USING CUST-REC,
313 CUSTOMER-BALANCE-STATS
314 MOVE CUST-ID TO RPT-CUST-ID
315 MOVE CUST-NAME TO RPT-CUST-NAME
316 MOVE CUST-OCCUPATION TO RPT-CUST-OCCUPATION
LOG 0 +---1---+---2---+---3---+---4---+---5--- LINE: 58 OF 60
0058 LIST AT ;
0059 The STATEMENT SAM1 ::> SAM1 :> 258.1 breakpoint action is:
0060 ;
PF 1: ? 2:STEP 3:QUIT 4:LIST 5:FIND 6:AT/CLEAR
PF 7:UP 8:DOWN 9:GO 10:ZOOM 11:ZOOM LOG 12:RETRIEVE
```

33

IBM Debug Tool for z/OS tutorial

© 2012 IBM Corporation

A "LIST AT" command is entered, and a list of breakpoints is shown in the log. Notice that now there is only one breakpoint set.

A CLEAR AT command clears all breakpoints



The screenshot displays the IBM Debug Tool interface. At the top, the command line shows `=> CLEAR AT; LIST AT` with a yellow box labeled "Enter" pointing to the prompt. A tip box states: "Tip: You can enter multiple commands by separating them with semi-colons". Below this, the source code for subroutine SAM2 is shown, with line 314 highlighted in red: `MOVE CUST-ID TO RPT-CUST-ID`. The log window at the bottom shows the execution of the commands: `0061 CLEAR AT ;`, `0062 LIST AT ;`, and `0063 There are no breakpoints set.`. A yellow oval highlights these log entries. The bottom of the screen shows the page number 34, the text "IBM Debug Tool for z/OS tutorial", and the copyright notice "© 2012 IBM Corporation".

You can quickly remove all of the breakpoints with a "CLEAR AT" command. You can also enter more than one command at a time, by separating commands with semi-colons. The commands "CLEAR AT; LIST AT" are typed on the command line, and Enter is pressed. That removed all remaining breakpoints. In the log, you can see that now there are no breakpoints set.

Commands to set statement breakpoints



- AT *statement-number* , or
- The default **F6** key (with the cursor placed on a statement), or
- A or AT line command in the source window
 - Sets a statement breakpoint
 - The program will stop when the statement is reached with GO or RUNTO commands
 - Frequently used command options:
 - AT FROM *n statement-number* sets a 'count' breakpoint that will stop the *n*th time the statement runs
 - For example, AT FROM 910 55 sets a breakpoint that will stop the 910th time, and each subsequent time, that statement 55 runs. It will not stop the first 909 times.
 - AT (*stmt-1, stmt-2, ..., stmt-n*) sets breakpoints at multiple statements
 - AT LABEL *name* sets a breakpoints at the named paragraph, section, or label

35

IBM Debug Tool for z/OS tutorial

© 2012 IBM Corporation

You have seen three ways to set statement breakpoints. The “AT statement number” command on the command line, an A or AT line command next to a statement, or by using the F6 key to set a breakpoint at a statement selected by your cursor. All of these methods have the same result. Use whichever method or methods you prefer.

There are some other commands you might find helpful for setting statement breakpoints. First, you can set a breakpoint that does not stop every time, but instead stops the *n*th time a statement runs. For example, the command "AT FROM 910 55" sets a breakpoint that will stop the 910th time, and each subsequent time, that statement 55 runs.

You can also set multiple statement breakpoints with one command, by entering multiple statement numbers within parentheses.

And an "AT LABEL" breakpoint lets you set a breakpoint at a COBOL paragraph or section name, or at a PLI or C label.

Commands to clear and list statement breakpoints



- **CLEAR AT** *statement-number* , or
- The default **F6** key (with the cursor placed on a statement with a breakpoint), or
- **C** line command in the source window
 - Clears a statement breakpoint
 - Frequently used command options:
 - **CLEAR AT** removes all breakpoints
 - **CLEAR AT** *start - end* clears all statement breakpoints from statement numbers *start* through *end*, inclusive
 - **CLEAR AT** (*stmt1, stmt2, ...*) clears breakpoints at multiple statements
 - **CLEAR AT LABEL** *name* clears the breakpoint at the named paragraph, section, or label

You can clear a statement breakpoint with a CLEAR AT command, with a C line command, or with the F6 key to clear the breakpoint at a cursor selected statement. And there are some other things you can do with clear commands.

If you enter the command: "CLEAR AT", all of your breakpoints will be removed. That's the easiest way to get rid of all of your breakpoints.

You can also remove all breakpoints in a range of statements. For example, the command "CLEAR AT 200 – 400" will remove any breakpoints that exist from statements 200 through 400, inclusive.

The next example shows that you can clear multiple statements with one command, by entering multiple statement numbers within parentheses.

And finally, label breakpoints are removed with "CLEAR AT LABEL ..." commands.

Commands to clear and list statement breakpoints



- **LIST AT** displays a list of breakpoints in the log
- **FINDBP** positions the source to the next statement breakpoint
 - Frequently used command options:
 - **FINDBP** finds the next statement breakpoint after the current cursor position in the source window
 - **FINDBP PREV** finds the previous statement breakpoint
 - **FINDBP FIRST** finds the first statement breakpoint
 - **FINDBP LAST** finds the last statement breakpoint

Use a "LIST AT" command to display a list of all current breakpoints in the log.

A FINDBP command is an easy way to see where breakpoints are in the source window. Enter FINDBP, and the source window is searched, and positioned to the next statement breakpoint. You can use the command modifiers PREVIOUS, first, and last, to control the search.

That is the end the first of three sections that describes how to use the debugger. Continue with the second part.

Feedback



Your feedback is valuable

You can help improve the quality of IBM Education Assistant content to better meet your needs by providing feedback.

- Did you find this module useful?
- Did it help you solve a problem or answer a question?
- Do you have suggestions for improvements?

Click to send email feedback:

mailto:iea@us.ibm.com?subject=Feedback_about_DTv12s12UsingTheDebuggerPart1.ppt

This module is also available in PDF format at: [../DTv12s12UsingTheDebuggerPart1.pdf](http://DTv12s12UsingTheDebuggerPart1.pdf)

You can help improve the quality of IBM Education Assistant content by providing feedback.



Trademarks, copyrights, and disclaimers

IBM, the IBM logo, ibm.com, z/OS, and zSeries are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of other IBM trademarks is available on the web at "[Copyright and trademark information](http://www.ibm.com/legal/copytrade.shtml)" at <http://www.ibm.com/legal/copytrade.shtml>

Other company, product, or service names may be trademarks or service marks of others.

THE INFORMATION CONTAINED IN THIS PRESENTATION IS PROVIDED FOR INFORMATIONAL PURPOSES ONLY. WHILE EFFORTS WERE MADE TO VERIFY THE COMPLETENESS AND ACCURACY OF THE INFORMATION CONTAINED IN THIS PRESENTATION, IT IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED. IN ADDITION, THIS INFORMATION IS BASED ON IBM'S CURRENT PRODUCT PLANS AND STRATEGY, WHICH ARE SUBJECT TO CHANGE BY IBM WITHOUT NOTICE. IBM SHALL NOT BE RESPONSIBLE FOR ANY DAMAGES ARISING OUT OF THE USE OF, OR OTHERWISE RELATED TO, THIS PRESENTATION OR ANY OTHER DOCUMENTATION. NOTHING CONTAINED IN THIS PRESENTATION IS INTENDED TO, NOR SHALL HAVE THE EFFECT OF, CREATING ANY WARRANTIES OR REPRESENTATIONS FROM IBM (OR ITS SUPPLIERS OR LICENSORS), OR ALTERING THE TERMS AND CONDITIONS OF ANY AGREEMENT OR LICENSE GOVERNING THE USE OF IBM PRODUCTS OR SOFTWARE.

© Copyright International Business Machines Corporation 2012. All rights reserved.